Claims

1. A software system for debugging a distributed software environment, the software system comprising:

a primary processing element having a software program that generates a corresponding event record in response to a selected event; and

a communication channel that links the primary processing element to a debugging host.

2. A software system according to claim 1 wherein the software program comprises:

at least two components;

a coordinator that manages control and data flow interactions between the components;

an interface between the coordinator and the component, the interface having a port that exposes an event;

a runtime system that collects the event record; and

a primary runtime debugging architecture that receives the event record from the runtime system and forwards the event record to the debugging host along the communication channel.

3. A software system according to claim 2 wherein the primary runtime debugging architecture comprises:

a time stamper to provide a time stamp to the event record generated by the software program; and

a causality stamper to provide an identification of a cause of the event associated with the corresponding event record.

4. A software system according to claim 3 wherein the primary runtime debugging architecture further comprises:

a primary uplink component to enable communication between the primary processing element and the debugging host along the communication channel; and

a primary transfer component coupled to the uplink component to collect and transfer the event record from the primary processing element to the debugging host along the communication channel.

5. A software system according to claim 3 wherein the software system further comprises:

an intermediate processing element disposed along the communication channel between the primary processing element and the debugging host to enable communication with the debugging host; and

the primary runtime debugging architecture further comprises a primary uplink component to enable communication between the primary processing element and the intermediate processing element.

6. A software system according to claim 5 wherein the intermediate processing element comprises;

an intermediate uplink component to enable communication between the intermediate processing element and the debugging host along the communication channel; and

an intermediate transfer component coupled to the intermediate uplink component to collect and transfer the event record from the intermediate processing element to the debugging host along the communication channel.

7. A software system according to claim 3 wherein the primary runtime debugging architecture further comprises a flash driver for interfacing with a flash memory to facilitate collection of the event record from the primary processing element and storage of the event record for subsequent transfer to a remote debugging host.

8. A software system according to claim 1 wherein the distributed software environment implements a predetermined design model having an explicitly defined event; and the software program generates the event record in response to an occurrence of the explicitly defined event.

9. A software system according to claim 1 wherein the distributed software environment comprises an explicit event recording call; and the software program generates the event record in response to an occurrence of the explicit event recording call.

10. A software system according to claim 1 wherein:

the distributed software environment executes on a target hardware platform; and

the target hardware platform comprises a probe for monitoring a selected bus trace on the target hardware platform and for generating an event record responsive to a predetermined activity on the bus trace.

11. A software system according to claim 1 wherein the software program generates a token representative of a predetermined sequence of events.